Gravitational Wave Data Analysis : Glitch classification using ML

Name:	**Meet Manoj Maratha**
Registration No./Roll No.:	19190
Institute/University Name:	IISER Bhopal
Program/Stream:	Data Science and Engineering
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

1 Introduction

The data provided contains information about the type of gravitational glitch a non-Gaussian noise transient is detected. A high occurrence rate for these glitches in data can endanger the data we use for finding gravitational waves. There are 22 types of glitches in the data, for which we are trying to prepare a model that can classify these glitches. Figure 1 displays the count plot of all the classes. Here we can see that some of the glitches have a considerably low count in our data. The data is skewed for specific attributes which are noticeable in the Table 1. When the maximum value of an attribute increases rapidly from its 75% value it represents skewness (we have highlighted such values).

Some machine learning algorithms prefer data to be not skewed as it can make its prediction worse. So to counter it, we scale our data using multiple scaling techniques to check which one gives us the best results. We used Robust Scaler, Quantile Transform (Uniform Distribution), and Quantile transform (Normal Distribution) as our scaling functions. As all these scalers handle outliers admirably, we choose them as it is unclear if there are any outliers in the data.

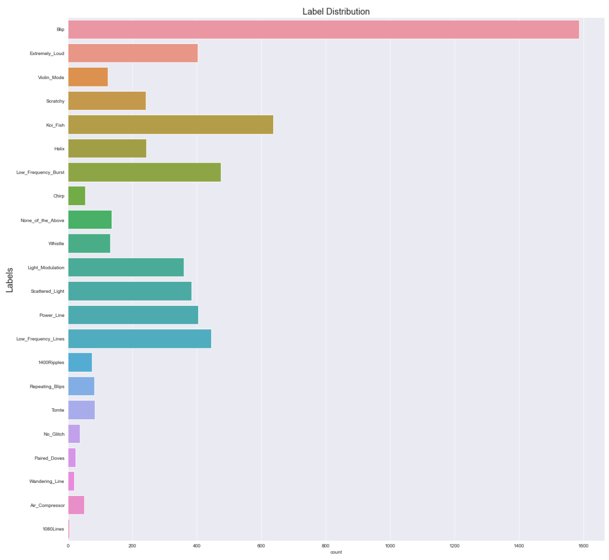


Figure 1: Countplot of different classes

Table 1: Description of data

	GPStime	centralFreq	peakFreq	snr	bandwidth	duration
count	6000	6000	6000	6000	6000	6000
mean	1.13e+9	1527.44	204.57	187.41	2937.87	1.78
std	3.17e+06	1319.21	375.03	1488.28	2663.56	2.68
min	1.13e+0.9	9.78	10.07	7.50	1.26	0.01
25%	1.129e+0.9	256.50	34.18	10.36	425.75	0.22
50%	1.132e+0.9	1228.92	111.13	15.43	2287.86	0.77
75%	1.135e+0.9	2627.85	183.50	37.03	5195.88	2.15
max	1.137e+0.9	4615.13	**2047.11**	**81178.73**	7946.48	**42.16**

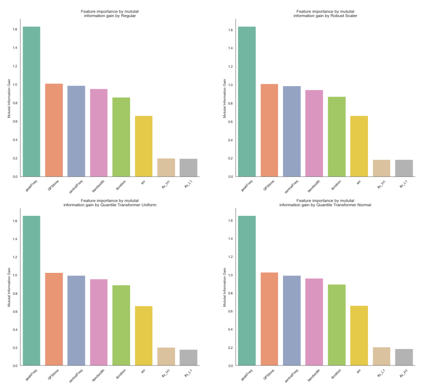
2 Methods

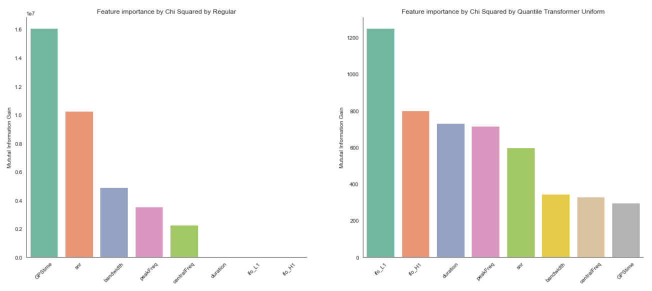
2.1 Data Preprocessing

We need to perform preprocessing on the data before used for building a machine learning model. We have a column named '*ifo*' that contains categorical data. One-Hot Encoding is the best way to encode information containing a considerably low number of unique values. In One-Hot Encoding, creates a vector with a length equal to the number of unique values and sets what value appeared as one. It prevents our machine learning model from being biased to a value.

2.2 Feautre Selection

Feature selection is a process of selection of a subset of attributes. We do this as it does not affect the accuracy of our model much and reduces the cost of computation. Two principal methods implemented in SKlearn that we can use for finding the importance of each attribute are '*mutual_info_classif*' and '*chi2*', which are the functions that calculate Mutual Information [1] of the discrete target variables in the data.


(a) Mutual Info classification



(b) Chi Squared

Figure 2: Importance for different scalers

Mutual information between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero only if two random variables are independent, and higher values mean high dependency. It depends on a nonparametric method based on entropy estimation from K-Nearest Neighbours distances as described in [1] and [2] which are established on the idea proposed in [3]. The importance of each attribute found using '*mutual_info_classif*' is displayed in Figure 2a.

Chi-Squared is a score that calculates chi-squared statistics from the data. The chi-squared test measures dependence between stochastic variables, so using this function highlights features that are

highly likely to be independent of class and hence can be dropped. As it can only be applied for positive values we do not apply it on Quantile Transform (Normal Distribution) and RObust Scaler. The importance of each attribute found using '*chi2*' is displayed in Figure 2b.
As we notice that the One-Hot Encoded ***ifo*** column is one of the least important attributes in *mutual_info_classif* whereas *chi2* says that it is one of the most important attributes. As they both contradict one other, we decided not to drop it. Hence no columns are dropped from the data.

2.3 Train Test Splitting

First, we drop the *id* attribute from the data as it is just an index for each datapoint. As there are some classes with a considerably low count in the data, we cannot use the simple *train_test_split* function from SKLearn with stratifying option for this data. Thus we use *StratifiedKFold* for splitting the data into five splits, dividing our data in 80-20 fashion, with 80% for training the model.

2.4 Model Training

We train and test the model on all the stratified data created. For hyperparameter tuning, we use *GridSearchCV* function provided in the SKLearn module.

2.4.1 Random Forest

We perform random forest classification on datasets that have scaled and regular data. In Random Forest Classifier [4] we make '*n_estimatores*' number of decision trees which have a subset of attributes selected from all the features. We then pass data from each of these decision trees and use averaging to improve the predictive accuracy. Table 2 exhibits tuning parameters and values.

Table 2: Random Forest Hyperparameter tuning parameters

Parameter Name	Parameter values		
criterion	*gini*	*entropy*	-
max_features	*none*	*sqrt*	*log2*
n_estimators	*10*	*100*	*1000*

2.4.2 Support Vector Clasifier

We perform Support Vector classification on datasets that are scaled. We do not use regular data here as Support Vector machine kernels require data to be centered around zero and have variance in the same order. Table 3 exhibits parameters to be tuned and their values. Note that the parameter '*degree*' is used only when the kernel is '*poly*'. Due to diminished combinations, we can supply more values for other parameters of the support vector classifier.

Table 3: Support Vector Classifier Hyperparameter tuning parameters

Parameter Name	Parameter values				
kernel	*linear*	*rbf*	*poly*	-	-
degree	*2*	*3*	-	-	-
C	*0.01*	*0.1*	*1*	*10*	*50*

2.4.3 Bagging Clasifier

We perform Bagging classification on datasets that have been scaled and normal. The bagging classifier uses a decision tree classifier as its base. This algorithm has several variations of this method in the literature. Samples drawn with replacement, such as what we are doing, it is known as Bagging [5]. Table 4 exhibits parameters to be tuned and their values.

Table 4: Bagging Classifier Hyperparameter tuning parameters

Parameter Name	Parameter values		
max_features	*0.7*	*1.0*	-
n_estimators	*10*	*100*	*1000*
max_samples	*0.7*	*1.0*	-

The code for the whole project is uploaded at GitHub

3 Evaluation Criteria

F1 score is an evaluation metric for classification, defined as the harmonic mean of precision and recall. Precision is the proportion of True positive to the total number of true positive and false positive. Recall is what proportion of true positives by the sum of true positives and false negatives for a class. F1 Macro averaged is the unweighted mean of the F1 score of each category. It is a helpful metric in unbalanced class distribution.

4 Analysis of Results

Table 5 contains analysis results of all the machine learning techniques used with different scalers. We are mentioning the score of models which performed best from all the splits performed by ***Stratified-KFold***. We can see that Bagging Classifier with Quantile Transform or Quantile Normal transform predicts classes of glitches with high F1 macro score. We can also see that the Bagging classifier is the best machine learning method that we can use for the given data, followed by Random Forest.

Table 5: Maximum F1 Macro Score of machine learning techniques on Stratified data

Model Name	Scaler			
	Regular	Robust Scaler	Quantile Transform	Quantile Normal Transform
Random Forest	97.0776%	97.0656%	97.0304%	97.0698%
Support Vector Classifier	-	64.4980%	84.2812%	83.0661%
Bagging Classifier	97.2164%	97.1950%	**97.2470%**	**97.2470%**

5 Discussions and Conclusion

A bagging classifier using decision trees is the best model for predicting glitches in gravitational wave data when the data is scaled using Quantile Transform. Exploration of other tree classifiers for better accuracy. Also, another type of scaler can be applied.

References

[1] Kraskov A. Estimating mutual information. *Physical Review E*, 69, 2004.

[2] Brian C. Ross. Mutual information between discrete and continuous data sets. *PLoS ONE*, 9, 2014.

[3] L. F. Kozachenko. Sample estimate of the entropy of a random vector. *Probl. Peredachi Inf.*, 23(2):9–16, 1987.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.